**21. 아키텍처 패턴의 개요**

전형적인 해결 방식 또는 예제를 의미, 기본적인 윤곽 제시.

장점 : 개발시간 단축, 고품질의 소프트웨어 생산, 안정적인 개발 가능, 의사소통 간편, 손쉽게 유지보수 수행 가능, 시스템 특성 예측 가능

|  |  |
| --- | --- |
| 패턴 | 설명 |
| 레이어 패턴  (Layers Pattern) | 서브 시스템의 계층 구조, 상위계층-서비스제공자, 하위계층-Client. 마주보는 두개의 계층 사이에서만 상호작용, 변경작업 용이  Ex)OSI 참조 모델(물리-데이터링크-네트워크….) |
| 클라이언트-서버 패턴  (Client-Server Pattern) | 서버 컴포넌트1, 다수 Client컴포넌트. 클라이언트-사용자 의사소통  서버는 항상 대기상태, 클라이언트와 서버는 서로 독립적 |
| 파이프-필터 패턴  (Pipe-Filter Pattern) | 데이터 스트림 절차의 각 단계를 필터 컴포넌트로 캡슐화하여 파이프를 통해 데이터를 전송하는 패턴 ↓  재사용성↑, 확장용이, 재배치하여 다양한 파이프라인 구축 가능  Ex)UNIX의 쉘(Shell) |
| 모델-뷰-컨트롤러 패턴  (Model-View-Controller Pattern) | 모델-기능과 데이터 보관, 뷰-정보표시, 컨트롤러-입력처리  대화형 어플리케이션에 적합 |

**22. 객체지향**

객체들을 조립해서 작성할 수 있는 기법. 소프트웨어의 재사용 및 확장 용이, 고품질의 소프트웨어를 빠르게 개발, 유지보수↑, 단계적, 계층적 표현, 병렬 처리 지원

|  |  |
| --- | --- |
| 객체(Object) | 데이터-객체의 정보 / 속성, 상태, 변수, 상수, 자료구조  함수-객체의 기능(알고리즘) / 메소드, 서비스, 동작, 연산 |
| 클래스(Class) | 공통된 속성과 연산을 가지는 객체의 집합, 타입을 의미  클래스에 속한 객체 = 인스턴스(Instance), 새로운 객체 생성 = 인스턴스화(Instantiation)  슈퍼클래스=상위(부모)클래스, 서브클래스=하위(자식)클래스 |
| 캡슐화(Encapsulation) | 세부 내용이 은폐되어 외부의 접근이 제한적. 외부 모듈의 변경으로 인한 파급 효과↓, 재사용 용이, Interface단순, 객체 간의 결합도↓ |
| 상속(Inheritance) | 상위 클래스의 모든 속성과 연산을 물려받는 것.  다중 상속 : 두개 이상의 상위 클래스로부터 속성과 연산 상속 |
| 다형성(Polymorphism) | 연산수행시 해당 객체(클래스)의 고유한 방법으로 응답할 수 있는 능력, 동일한 메소드 명을 사용하며 같은 의미의 응답을 함  Ex)+연산자 숫자클래스-덧셈, 문자클래스-문자열의 연결 |

**23. 모듈**

단독 컴파일 가능, 재사용o, 기능적 독립성(다른 모듈과의 과한 상호작용 배제), 독립성↑ 영향↓ 오류 해결↑, 하나의 기능만을 수행, 독립성↑ = 결합도(Coupling)↓, 응집도(Cohesion)↑, 크기↓

1. 결합도(Coupling)

모듈 간에 상호 의존 정도 or 연관관계 의미, 결합도↓, 품질↑

결합도↑, 시스템 구현 및 우지보수 작업 어려움

|  |  |
| --- | --- |
| 자료 결합도  (data Coupling) | 자료 요소로만 구성, 다른 모듈을 호출할 때 매개 변수나 인수로 데이터를 넘겨주고 처리결과를 돌려받음, 가장바람직함 |
| 스탬프 결합도  (검인, Stamp) | 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도.  결합도  ↓  결합도↑  두 모듈이 동일한 자료구조 조회하는 경우의 결합도.  자료구조의 변화는 조회하는 모든 모듈에 영향을 미침 |
| 제어 결합도  (Control) | 어떤 모듈이 다른 모듈 내부의 논리 흐름을 제어하기 위해 신호를 보냄  처리 기능이 두 모듈에 분리되어 설계된 경우에 발생  하위 모듈에서 상위 모듈로 제어 신호가 이동하여 하위 모듈이 상위 모듈에게 처리 명령을 내린다 <-권리 전도 현상 |
| 외부 결합도  (External) | 어떤 모듈에서 선언한 데이터를 외부의 모듈에서 참조할 때의 결합도  참조되는 데이터의 범위를 각 모듈에서 제한가능 |
| 공통 결합도  (공유, Common) | 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도  데이터 내용 변경시에 이를 사용중인 모든 모듈에게 영향을 끼침 |
| 내용 결합도  (Content) | 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도, 내부로 제어가 이동하는 경우에도 해당됨. |

2. 응집도(Cohesion)

정보 은닉 개념을 확장, 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미한다.

응집도↑

응집도↓

|  |  |
| --- | --- |
| 기능(Functional) | 모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우 |
| 순차적(Sequential) | 모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도 |
| 교환(통신)적  (Communication) | 동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도 |
| 절차적 응집도  (Procedural) | 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도 |
| 시간적(Temporal) | 특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우 |
| 논리적(Logical) | 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도 |
| 우연적(Coinciental) | 모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우 |

팬인(Fan-In) – 어떤 모듈을 제어(호출)하는 모듈의 수, 팬인↑ 재사용↑, 장애발생위험↑, 테스트필요

팬아웃(Fan-Out) – 어떤 모듈에 의해 제어(호출)되는 모듈의 수

**24. 공통 모듈**

1. 개요

여러 프로그램에서 공통적으로 사용할 수 잇는 모듈을 의미 ex)자주 사용되는 계산식, 사용자인증

재사용성+중복 개발 회피를 위해 공통부분 식별, 명확한 기능 이해를 이해 명세 기법 준수↓

정확성(Correctness),명확성(Clarity), 완전성(Completeness), 일관성(Consistency), 추적성(Traceability)

2. 재사용

비용, 개발시간 절약하기 위해 이미 개발된 기능들을 재구성하여 제작물에 최적화시키는 작업.

3. 효과적인 모듈 설계 방안

모듈의 독립성↑재사용성↑, 일관성 유지, 유지보수 용이, 예측 가능한 기능, 적당한 크기

**25. 코드**

1. 개요

자료를 처리하는 과정에서 분류, 조합 및 집계를 용이하게 하고, 특정 자료의 추출을 쉽게 하기 위해서 사용하는 기호 ex)주민등록번호, 학번

주요기능 – 식별 기능(데이터 성격), 분류 기능(그룹화), 배열 기능(나열)

2. 종류

|  |  |
| --- | --- |
| 순차(Sequence) | 자료의 발생 순서, 크기, 순서 등 일정 기준에 따라 차례로 일련번호 부여 |
| 블록(Block) | 공통성이 있는 항목들을 블록으로 구분 후 일련번호를 부여 |
| 10진(Decimal) | 10진분할 후 다시 10진분할 ex) 1000 : 공학, 1100 : 소프트웨어공학 |
| 그룹 분류  (Group Classification) | 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분 후 일련번호 부여  Ex) 1-01-001 : 본사-총무부-인사계, 2-01-001 : 지사-총무부-인사계 |
| 연상(Mnemonic) | 항목의 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 부여 |
| 표의 숫자 | 성질(크기, 부피, 지름 등)의 물리적 수치를 코드에 적용, 유효 숫자 코드라고도 한다. Ex)120-72-1500 = 두께x폭x길이 |
| 합성(Combined) | 2개 이상의 코드를 조합하여 만드는 방법 ex)KE-711(대한항공 711기) |

3. 코드 부여 체계 - 이름만으로 개체의 용도와 적용 범위를 알 수 있도록 코드를 부여하는 방식

**26. 디자인패턴**

1. 개요

모듈의 세부적인 구현 방안을 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제를 의미

한 패턴에 변형을 가하거나 특정 요구사항을 반영하면 유사한 형태의 다른 패턴으로 변화

※아키텍처 패턴과 디자인 패턴의 차이점 알아 두기

2. 생성 패턴(Creational Pattern)

추상팩토리, 빌더, 팩토리 메소드, 프로토타입, 싱글톤

3. 구조패턴(Structural Pattern)

어댑터, 브리지, 컴포지트, 데코레이터, 퍼싸드, 플라이웨이트, 프록시

4. 행위패턴(Behavioral Pattern)

책임 연쇄, 커맨드, 인터프리터, 반복자, 중재자, 메멘토, 옵서버, 상태, 전략, 템플릿메소드, 방문자

**27. 시스템 인터페이스 요구사항 분석**

시스템 인터페이스 - 독립적으로 떨어져 있는 시스템들끼리 서로 연동하여 상호 작용하기 위한 접속 방법이나 규칙을 의미한다.

요구사항 명세서에서 요구사항을 기능적 요구사항과 비기능적 요구사항으로 분류하고 조직화하여 요구사항 명세를 구체화하고 이를 이해관계자에게 전달하는 일련의 과정

※분석절차

선별 및 목록작성 – 자료를 준비 – 요구사항 분류 – 요구사항 분석 및 내용 추가, 수정 – 전달

**28. 인터페이스 요구사항 검증(Requirements Verification)**

인터페이스의 설계 및 구현 전 사용자들의 요구사항이 요구사항 명세서에 정확하고 완전하게 기술되었는지 검토하고 개발 범위의 기준인 베이스라인을 설정하는 것

요구사항 검토 계획 수립 → 검토 및 오류 수정 → 베이스라인 설정 순으로 수행

1. 인터페이스 요구사항 검토 계획 수립

검토 기준 및 방법, 참여자 선정, 체크리스트 작성(완전성 등), 관련 자료 준비, 검토 일정 확인

2. 인터페이스 요구사항 검토 및 오류 수정

검토 체크리스트의 항목에 따라 인터페이스 요구사항 명세서를 검토 → (오류 발견 시)오류 목록과 시정 조치서 작성 → 검토결과 전달 → (시정 조치서 작성시)완료 확인 및 검토 작업 완료

3. 인터페이스 요구사항 베이스라인 설정

검증된 요구사항은 프로젝트관리자와 주요 의사 결정자에게 공식적으로 승인 후 베이스라인 설정

4.요구사항 검증 방법

요구사항 검토(Requirements Review) : 오류 확인 및 표준 준수 여부 등의 결함 여부를 담당자들이 분석하는 방법, 동료검토(Peer Review), 워크스루(Walk Through), 인스펙션(Inspection)등이 있음

5. 인터페이스 요구사항 검증의 주요 항목

완전성(Completeness), 일관성(Consistency), 명확성(Unambiguity), 기능성(Functionality), 검증가능성(Verifiability), 추적 가능성(Traceability), 변경 용이성(Easily Cangeable)

**29. 인터페이스 시스템 식별**

1. 개발 시스템 식별

개발하고자 하는 시스템의 상세 식별 정보를 정의하고 목록을 작성하는 것

2. 내 ∙ 외부 시스템 식별

개발할 시스템과 연계할 내, 외부 시스템들의 상세 식별 정보를 정의하고 목록을 작성하는 것

3. 내 ∙ 외부 시스템 환경 및 관리 주체 식별

연계할 시스템 접속에 필요한 IP, URL, Port정보 등 시스템의 실제 운영 환경, 관리 주체-담당자

4. 내 ∙ 외부 시스템 네트워크 연결 정보 식별

시스템 로그인 및 DB 정보, 인터페이스 관련 자료들을 기반으로 네트워크 연결 정보 확인

5. 인터페이스 식별

인터페이스 요구사항 명세서와 인터페이스 요구사항 목록을 기반으로 개발할 시스템과 이와 연계할 내 ∙ 외부 시스템 사이의 인터페이스를 식별하고 인터페이스 목록을 작성하는 것

6. 인터페이스 시스템 식별

인터페이스에 참여하는 시스템들을 송신 시스템과 수신 시스템으로 구분하여 작성하는 것

**30. 송 ∙ 수신 데이터 식별**

1. 식별 대상 데이터

송, 수신 시스템 사이에서 교환되는 데이터, 규격화된 표준 형식에 따라 전송.

종류 – 인터페이스 표준 항목, 송 ∙ 수신 데이터 항목, 공통 코드

※인터페이스 표준항목

송 ∙ 수신 시스템을 연계하는데 표준적으로 필요한 데이터를 의미, 시스템 공통부 / 거래 공통부

송 ∙ 수신 데이터 항목 – 업무용 데이터, 공통 코드 -시스템의 공통적인 코드 ex)코드값, 코드명

2. 정보 흐름 식별

개발할 시스템과 내 ∙ 외부 시스템 사이에서 전송되는 정보들의 방향성을 식별하는 것

인터페이스 목록 확인해 정보 흐름 식별 → 교환되는 주요 데이터 항목이나 정보 그룹 도출

3. 송 ∙ 수신 데이터 식별

개발할 시스템과 연계할 내 ∙ 외부 시스템 사이의 정보 흐름과 DB 산출물을 기반으로 식별

인터페이스 표준 항목과 송 ∙ 수신 데이터 항목 식별 : 교환 범위 확인후 항목 식별

코드성 데이터 항목 식별 : 코드, 코드명, 코드 설명 등의 코드 정보를 식별한다.

**31. 인터페이스 방법 명세화**

1. 명세화의 개념

오류 식별 및 처리 방안에 대한 내용을 문서로 명확하게 정리하는 것

송 ∙ 수신 방법 명세화 – 시스템 연계 기술, 인터페이스 통신 유형, 처리 유형, 발생 주기 정보必

2. 시스템 연계 기술

개발할 시스템과 내 ∙ 외부 시스템을 연계할 때 사용되는 기술을 의미

DB Link, API/Open API, 연계 솔루션, Socket, Wep Service

3. 인터페이스 통신 유형

개발할 시스템과 내 ∙ 외부 시스템 간 데이터를 송 ∙ 수신하는 형태를 의미

단방향-거래 요청, 응답x, 동기–응답 대기(Request-Reply), 비동기–요청 후 다른작업, 응답 후 처리

4. 인터페이스 처리 유형

송 ∙ 수신 데이터를 어떤 형태로 처리할 것인지에 대한 방식을 의미

실시간 방식-바로 처리, 지연 처리 방식–비용이 많이 발생할 때, 배치 방식–대량의 데이터 처리

5. 인터페이스 발생 주기

개발할 시스템과 내 ∙ 외부 시스템 간 송 ∙수신 데이터가 전송되어 인터페이스가 사용되는 주기

6. 송 ∙ 수신 방법 명세화

연계 방식, 통신 및 처리 유형, 발생 주기 등의 송 ∙ 수신 방법 정의 및 명세 작성

7. 송 ∙ 수신 데이터 명세화

인터페이스 시 필요한 송 ∙ 수신 데이터에 대한 명세 작성

8. 오류 식별 및 처리 방안 명세화

발생할 수 있는 오류 식별 및 오류 처리 방안에 대한 명세 작성

**32. 시스템 인터페이스 설계서 작성**

1. 시스템 인터페이스 설계서의 개요

시스템이 갖는 인터페이스 목록과 각 인터페이스의 상세 데이터 명세를 정의한 문서(현황 확인)

송 ∙ 수신 방법, 데이터 명세화 과정에서 작성한 산출물을 기반으로 작성, 누락 보완 필요시 수정

2. 시스템 인터페이스 목록 작성

업무 시스템과 내 ∙ 외부 시스템 간 데이터를 주고받는 경우에 사용하는 인터페이스에 대해 기술

3. 시스템 인터페이스 정의서 작성

인터페이스 별로 시스템 간의 연계를 위해 필요한 데이터 항목 및 구현 요건 등을 기술하는 것

ID, 명, 처리 유형, 통신 유형, 주기, 구분, 포맷 – 송 ∙ 수신 데이터 명세 참조 – 매핑 규칙 작성

**33. 미들웨어 솔루션 명세**

1. 미들웨어(Middleware)의 개념 및 종류

운영체제와 응용 프로그램 사이에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공

표준화된 인터페이스를 제공 = 시스템 간의 데이터 교환에 일관성을 보장 (DB, RPC, MOM 등)

|  |  |
| --- | --- |
| DB(Data base) | 클라이언트에서 원격의 데이터베이스와 연결, 2-Tier 아키텍처  Ex); ODBC, IDAPI, Glue |
| RPC  (Remote Procedure Call) | 프로시저를 사용, 원격 프로시저를 로컬 프로시저처럼 호출  Ex) Entera, ONC/RPC |
| MOM(Message  Oriented Middleware) | 메시지 기반의 비동기형 메시지를 전달(MQ, Message Q, JMS)  이기종 분산 데이터 시스템의 데이터 동기를 위해 많이 사용 |
| TP-Monitor(Transaction Processing Monitor) | 온라인 트랜잭션 업무에서 트랜잭션을 처리 및 감시하는 미들웨어  사용자 수가 증가해도 빠른 응답 속도 유지해야함 (Tuxedo, tmax) |
| ORB  (Object Request Broker) | 객체 지향 미들웨어, 코바 표준 스펙 구현  Ex) Orbix, CORBA |
| WAS  (Web Application Server) | 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용  Ex) WebLogic, WebSphere |

8. 미들웨어 솔루션 식별

개발 및 운영 환경에 사용될 미들웨어 솔루션을 확인하고 목록을 작성

구매예정 소프트웨어 확인 후 미들웨어 솔루션 식별 – 솔루션 목록 작성 – 재확인 후 수정

9. 미들웨어 솔루션 명세서 작성

미들웨어 솔루션 목록에서 솔루션 별로 관련 정보들을 상세하게 기술하는 것

제품 정보 검토 – 사용 환경과 특징 검토 – 성능과 기능 검토 – 제약사항 정리 후 명세서 작성